# Demand-Aware Networks: Metrics and Algorithms

Chen Avin and Stefan Schmid

"We cannot direct the wind,
but we can adjust the sails."

(Folklore)

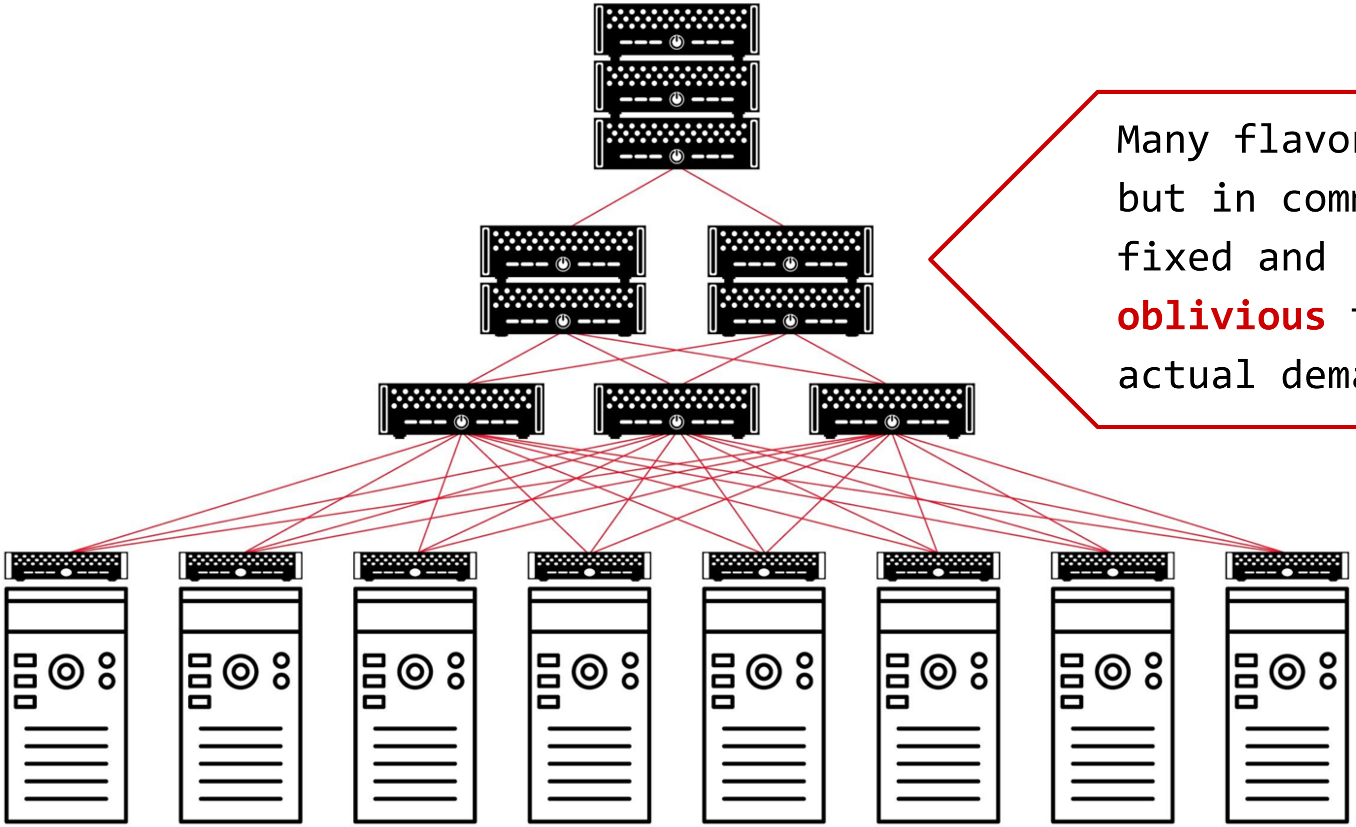
Acknowledgements:

German-Israeli Foundation for Scientific Research and Development

European Research Council
Established by the European Commission

# Today’s Datacenters

## Fixed and Demand-Oblivious Topology

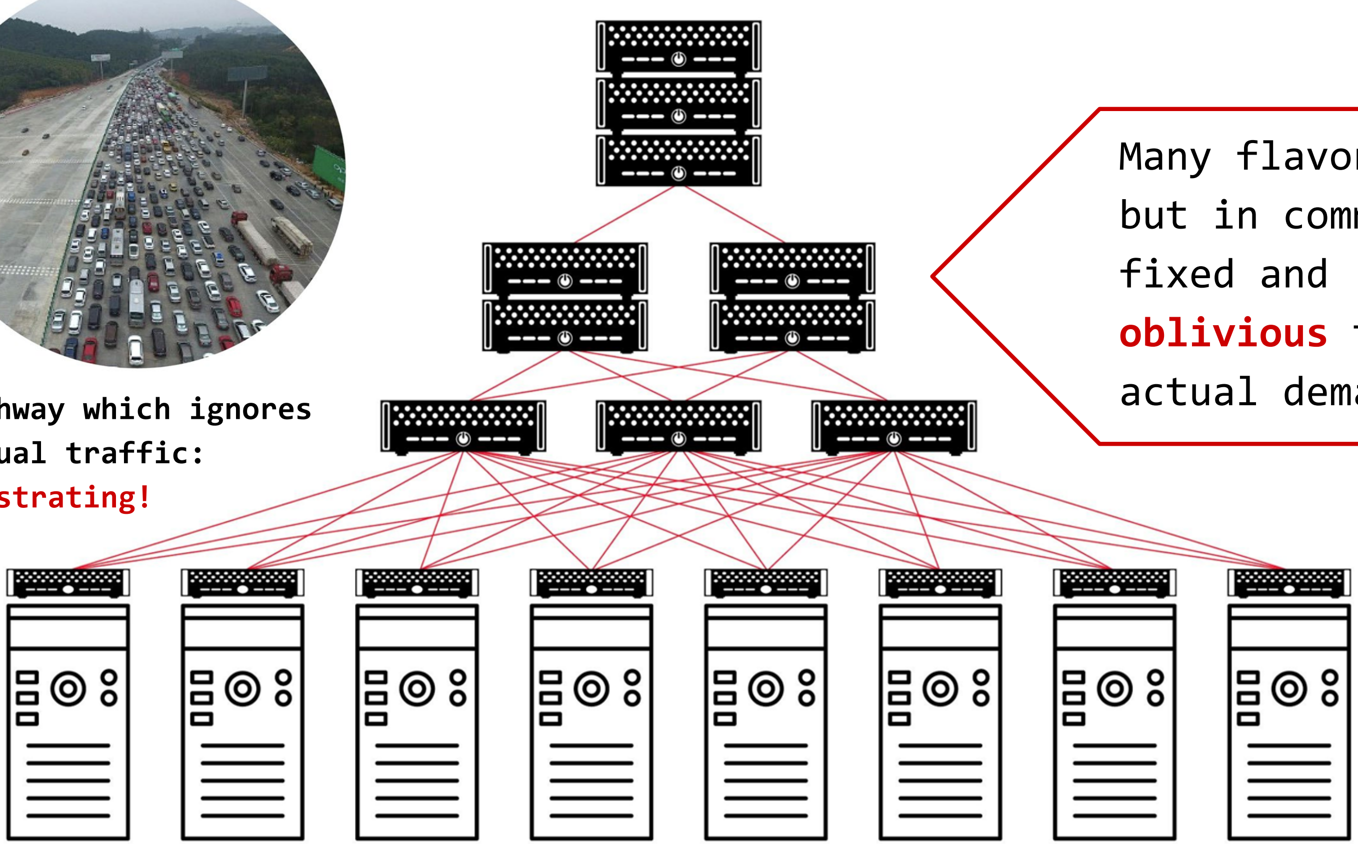


Many flavors, but in common: fixed and **oblivious** to actual demand.

# Today’s Datacenters

## Fixed and Demand-Oblivious Topology

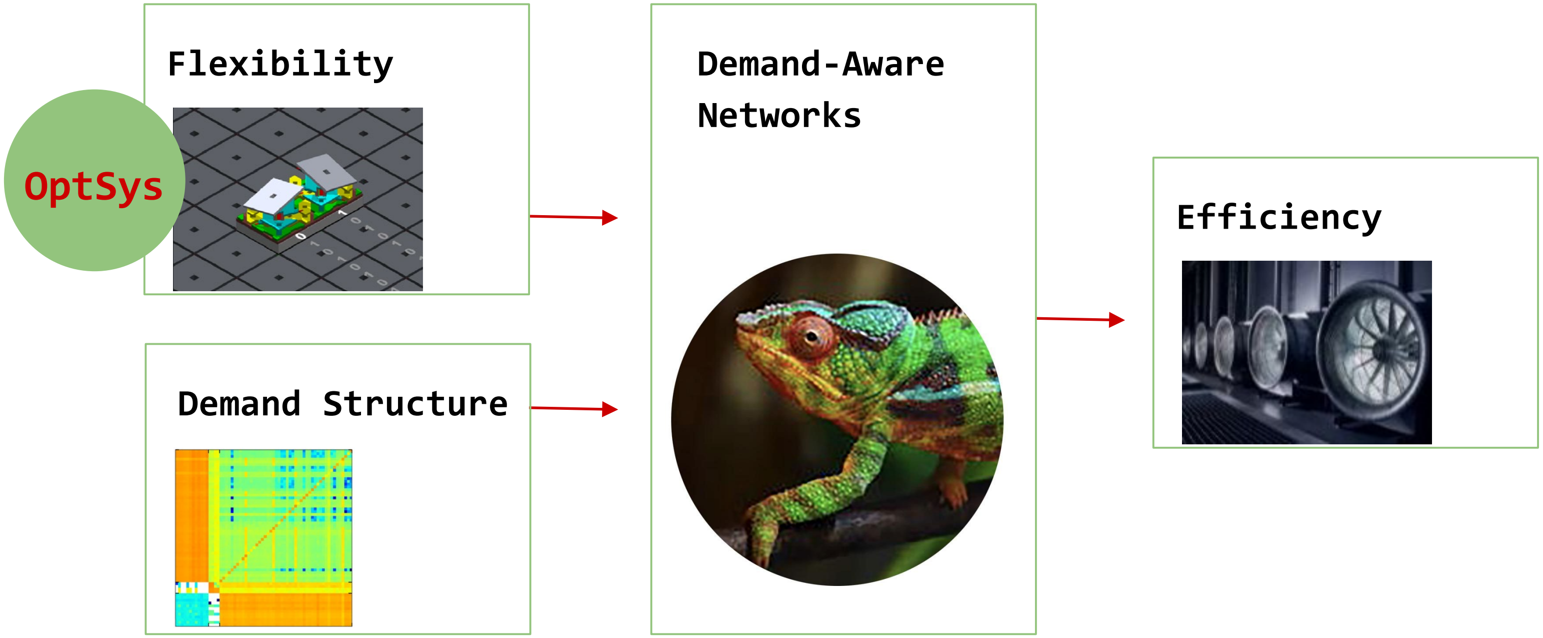
**Highway which ignores actual traffic: frustrating!**



Many flavors, but in common: fixed and **oblivious** to actual demand.

# Vision

## Demand-Aware Networks



OptSys

Flexibility

Demand Structure

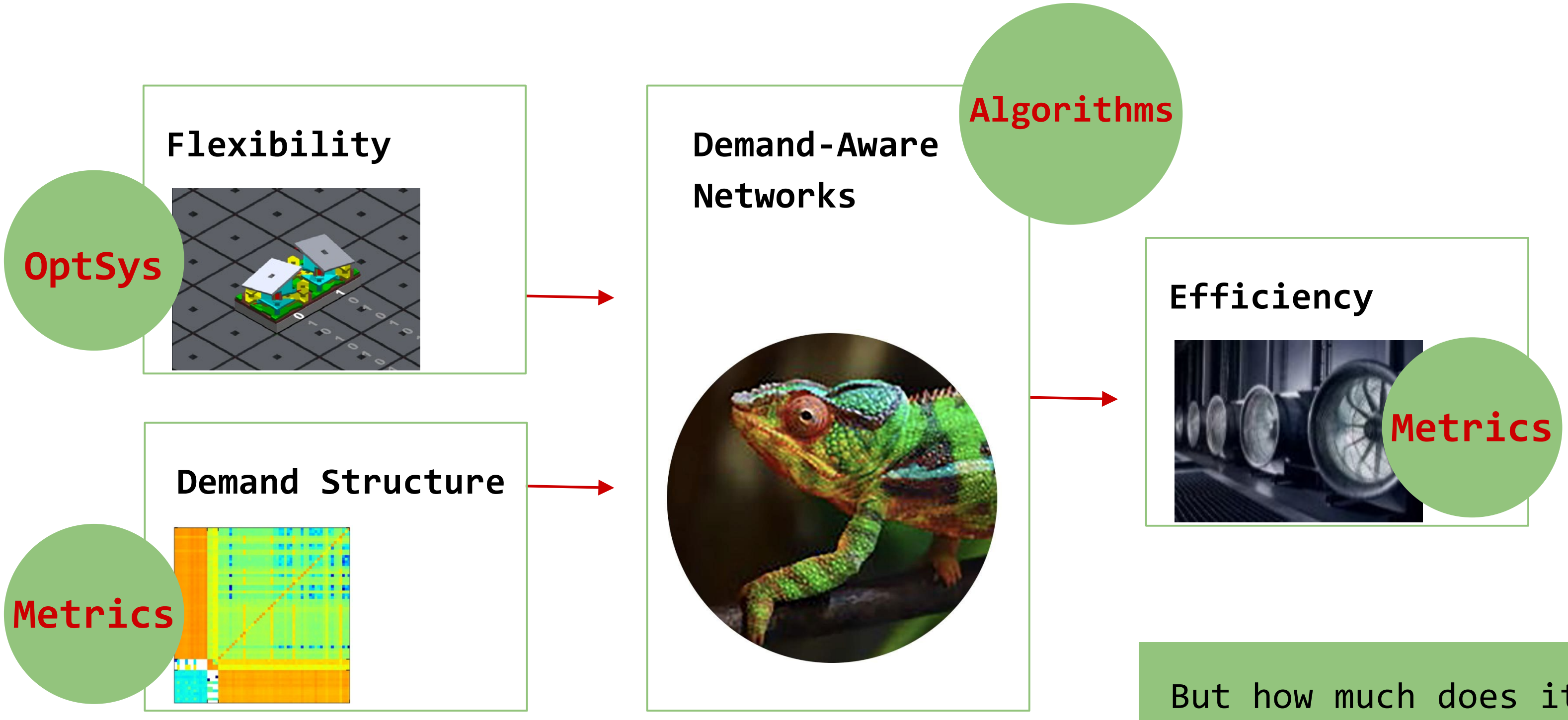
Demand-Aware Networks

Efficiency

Now is the time!

# Vision

Demand-Aware Networks



Flexibility

OptSys

Demand Structure

Metrics

Demand-Aware Networks

Algorithms

Efficiency

Metrics

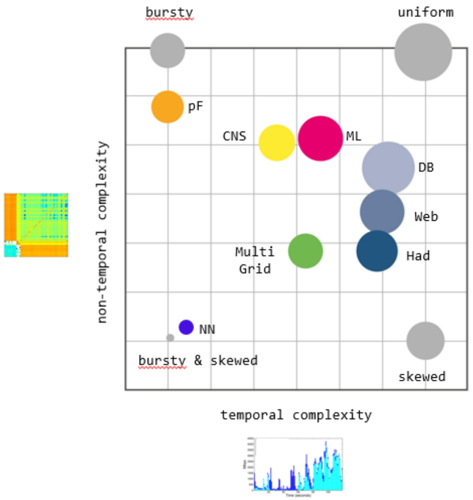
Now is the time!

But how much does it help? As usual in computer science: **it depends!** We need metrics for demand **structure** and for possible **efficiency**.

# Our Perspective

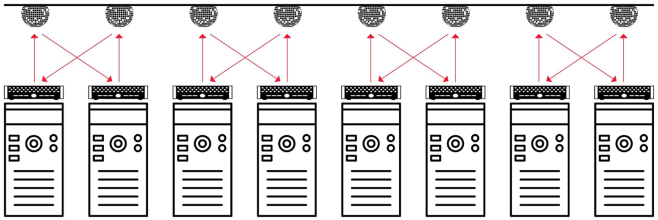
## Information Theory and Entropy

Demand entropy:
Spatial and temporal
**structure** of traffic



&

Entropy: A tight metric for
the achievable **route lengths**
in demand-aware networks



Question 1:

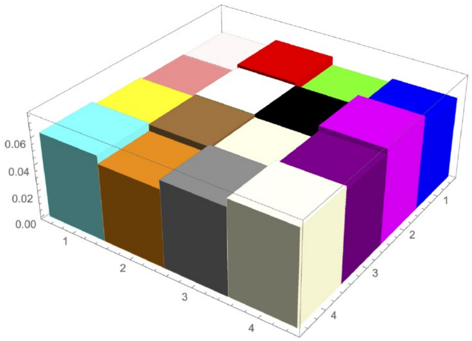
# How to Quantify such “Structure” in the Demand?

# Intuition

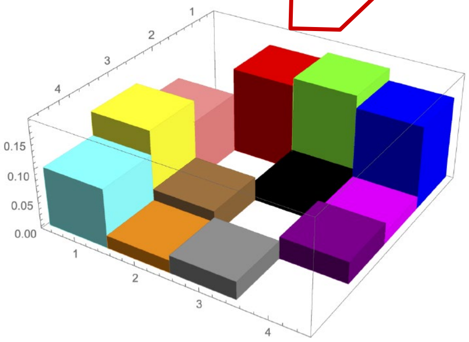
## Which demand has more structure?

⇢ Traffic matrices of two different distributed ML applications

- → GPU-to-GPU



VS



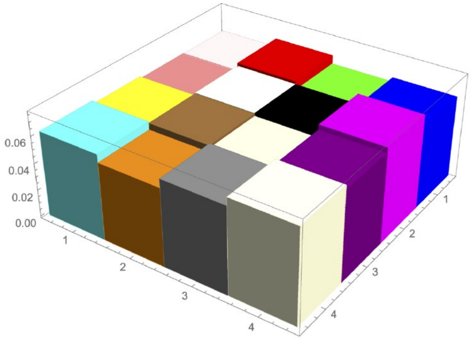
Color = communication pair

# Intuition

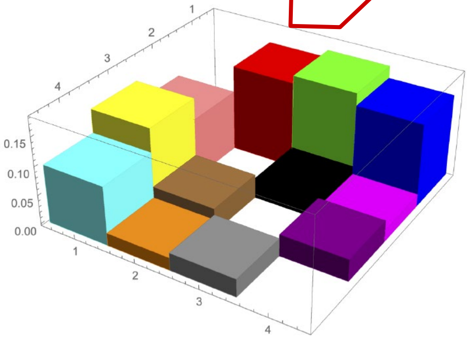
Which demand has more structure?

⇢ Traffic matrices of two different distributed ML applications

⇀ GPU-to-GPU



vs



Color = communication pair

**More uniform**

**More structure**

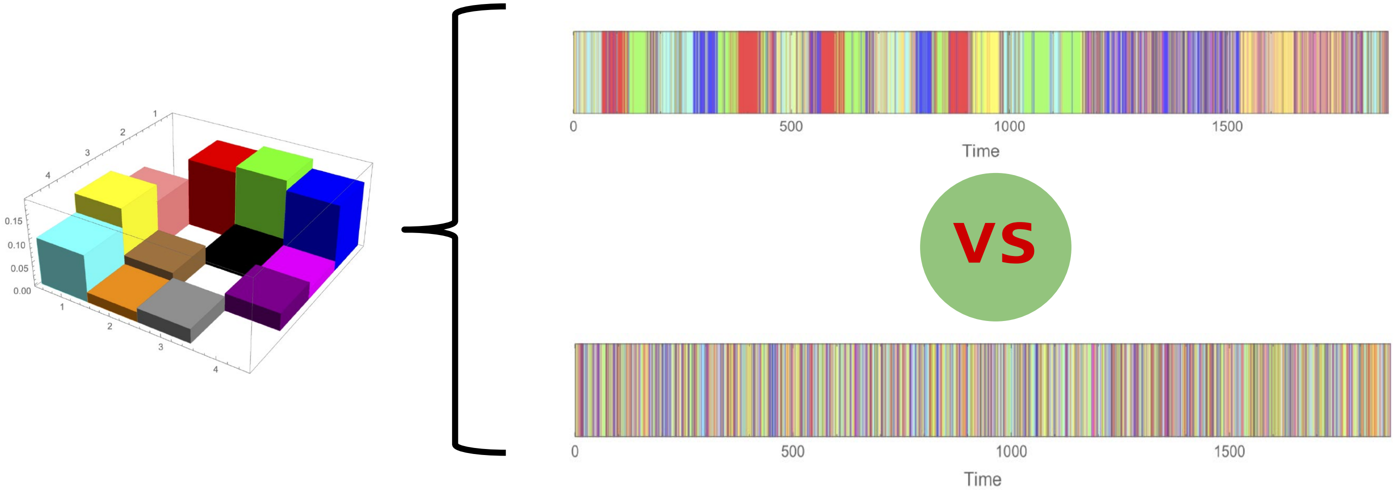
# Intuition

## Spatial vs Temporal Structure

⇢ Two different ways to generate same traffic matrix:

⇀ same non-temporal structure

⇢ Which one has more structure?



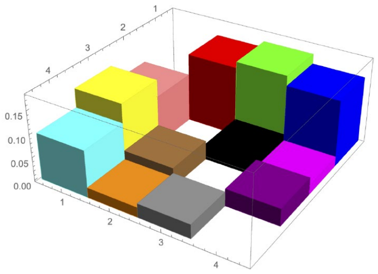
# Intuition

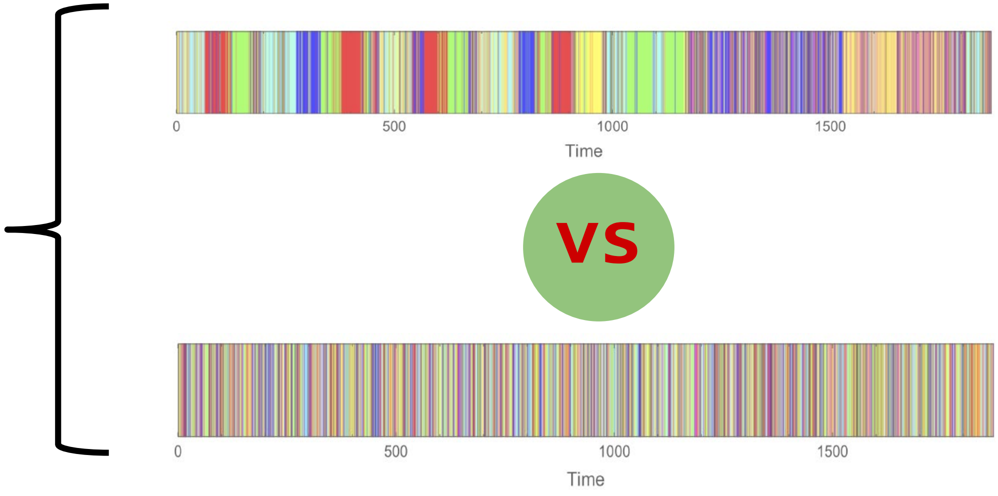
## Spatial vs Temporal Structure

⇢ Two different ways to generate same traffic matrix:

  ⇀ same non-temporal structure

⇢ Which one has more structure?



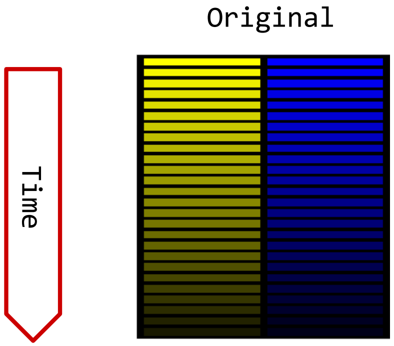


Systematically?

# Trace Complexity

Information-Theoretic Approach

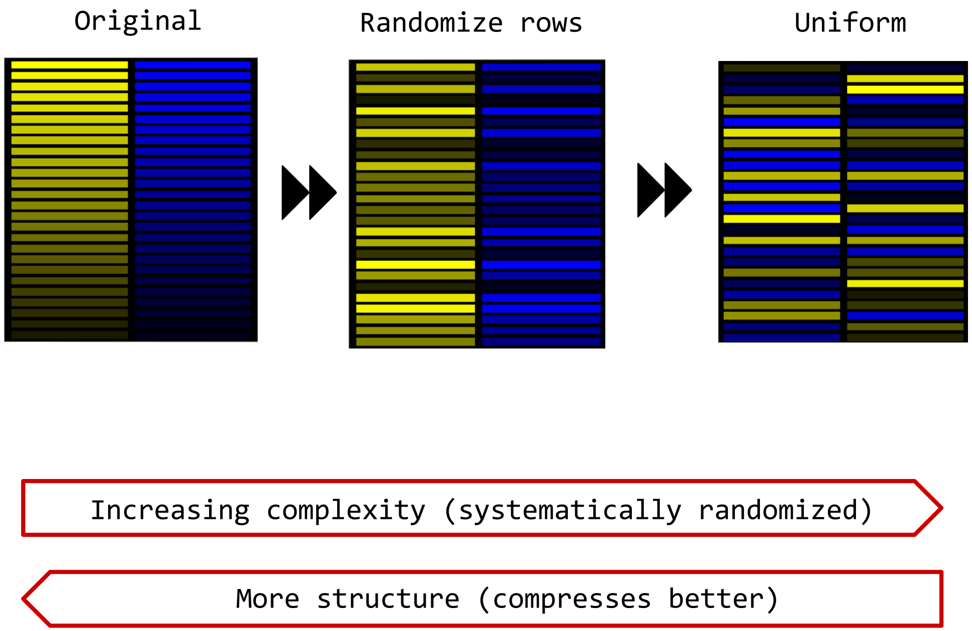
“Shuffle&Compress”



# Trace Complexity

## Information-Theoretic Approach

## “Shuffle&Compress”



Original

Randomize rows

Uniform

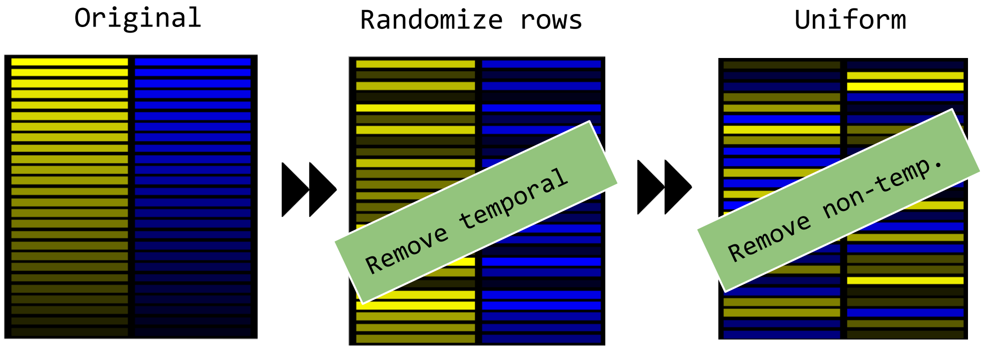
Increasing complexity (systematically randomized)

More structure (compresses better)

# Trace Complexity

## Information-Theoretic Approach

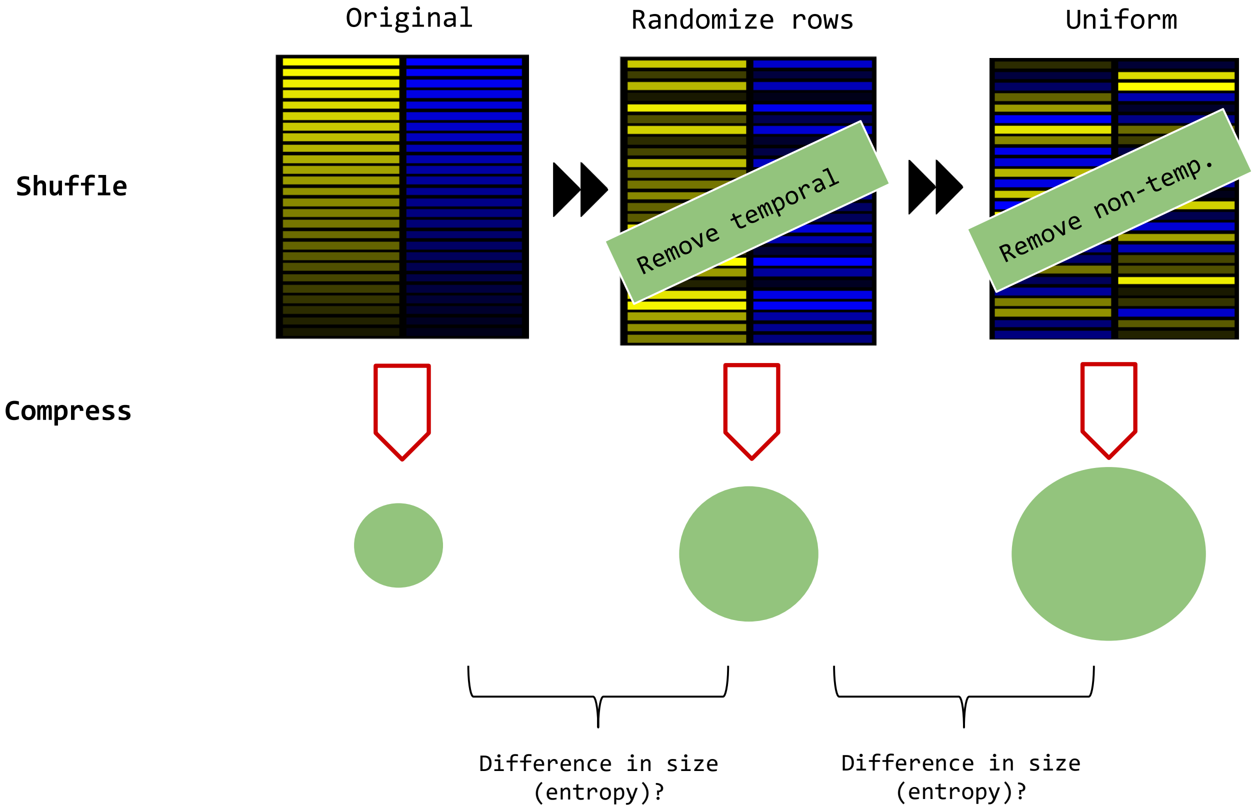
“Shuffle&Compress”



# Trace Complexity

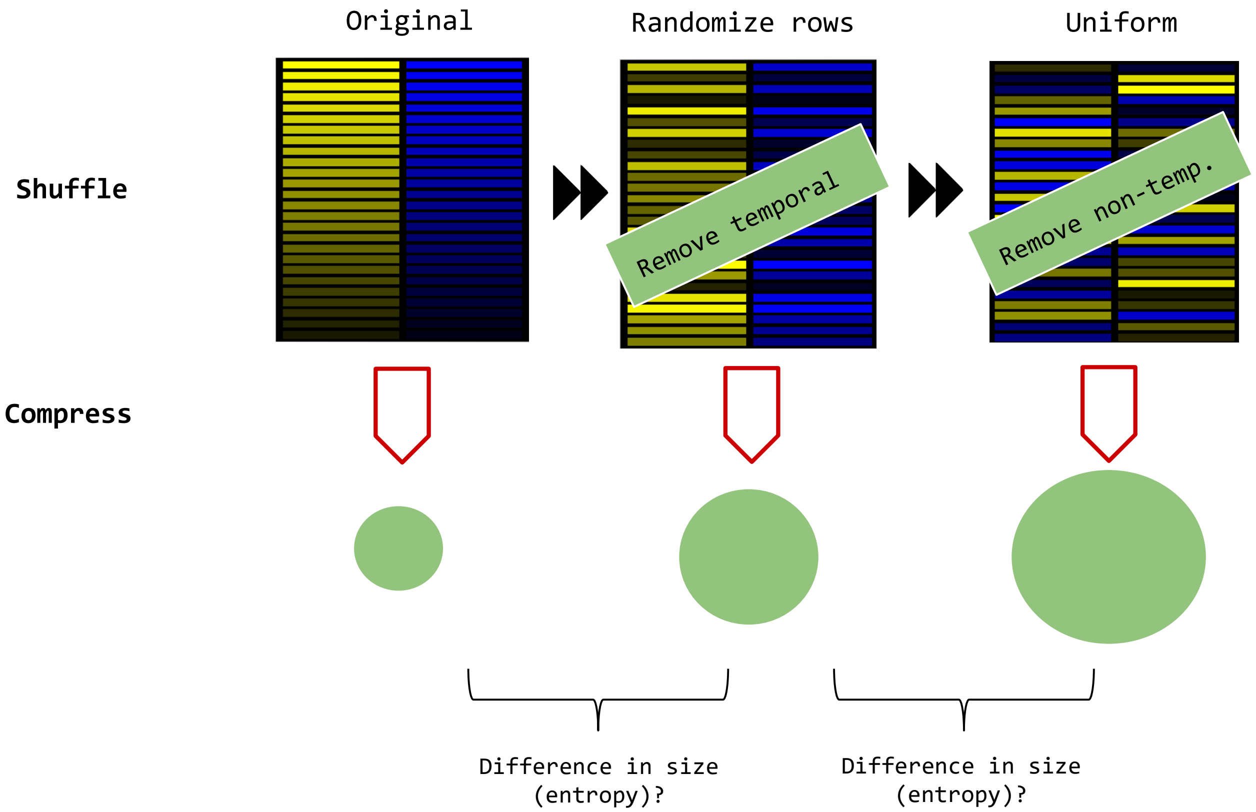
## Information-Theoretic Approach

## “Shuffle&Compress”



# Trace Complexity

## Information-Theoretic Approach
## “Shuffle&Compress”



Original

Randomize rows

Uniform

**Shuffle**

Remove temporal

Remove non-temp.

**Compress**

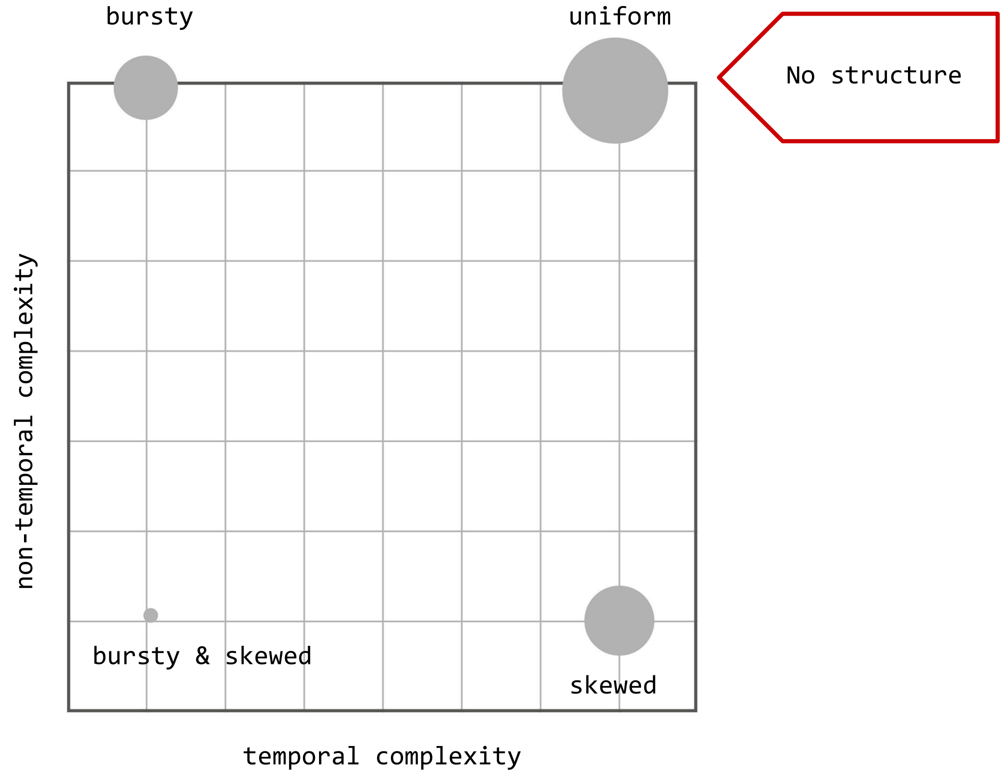
Difference in size (entropy)?

Difference in size (entropy)?

**Can be used to define 2-dimensional complexity map!**

Trace Complexity

# Complexity Map



bursty

uniform

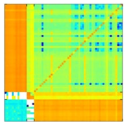
No structure

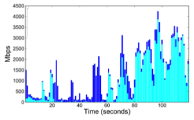
non-temporal complexity

bursty & skewed

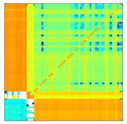
skewed

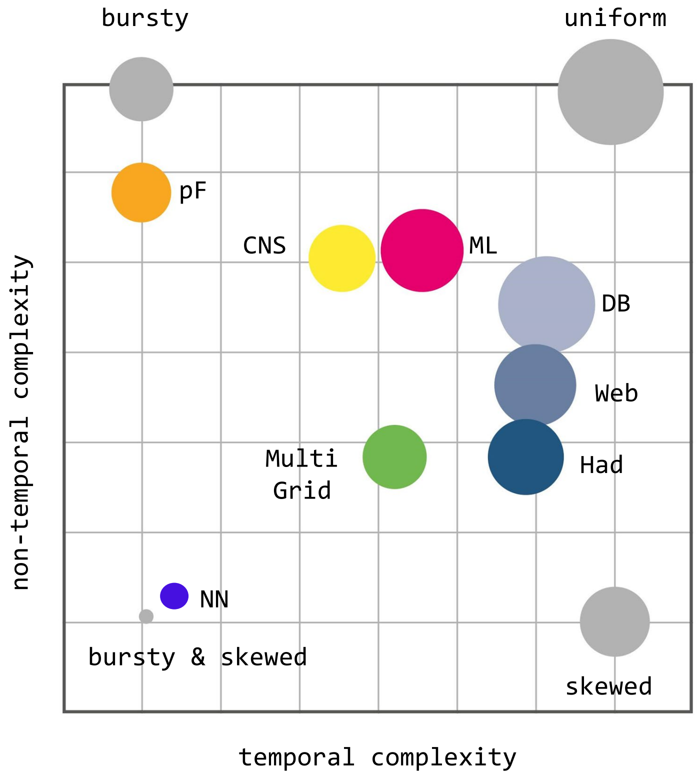
temporal complexity

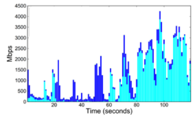




# Complexity Map





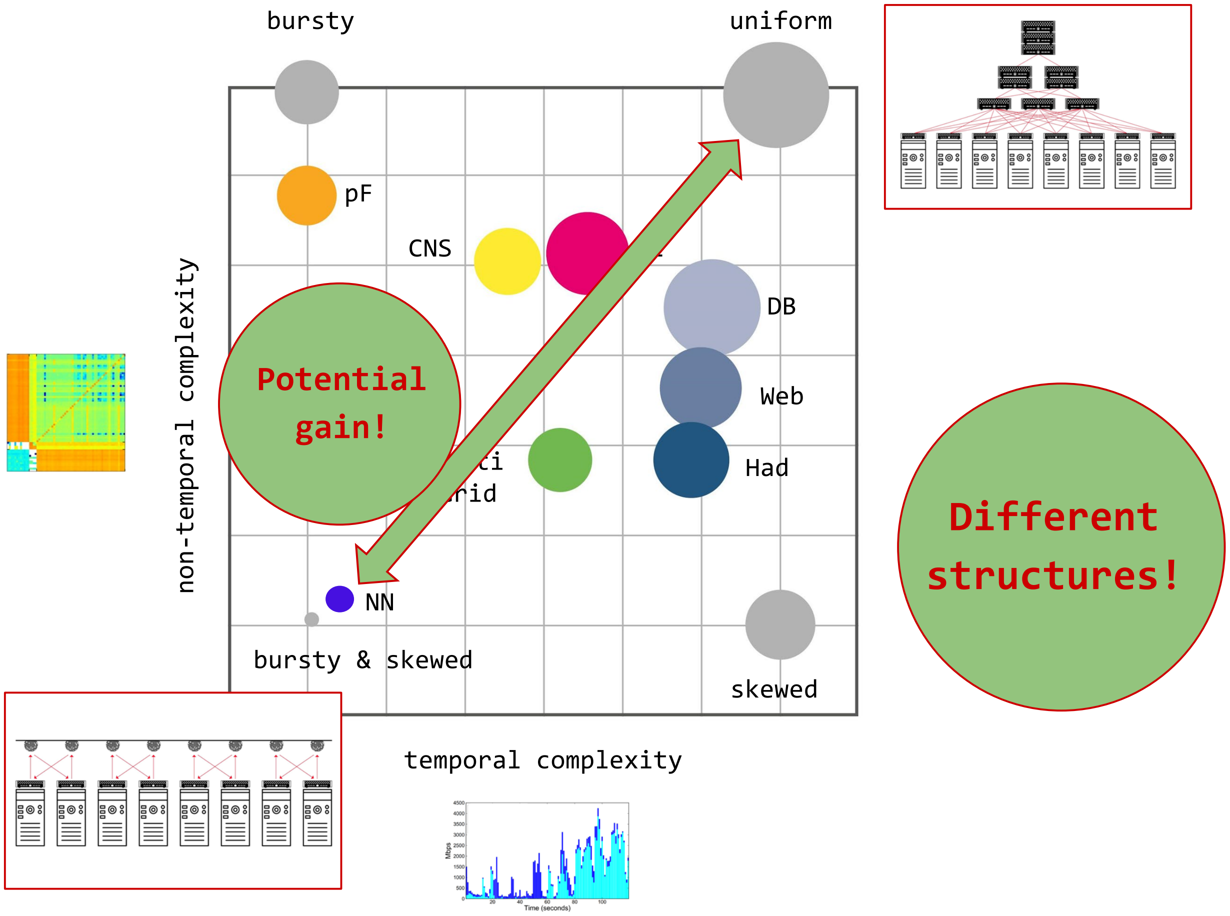


No structure

**Different structures!**

Trace Complexity

# Complexity Map

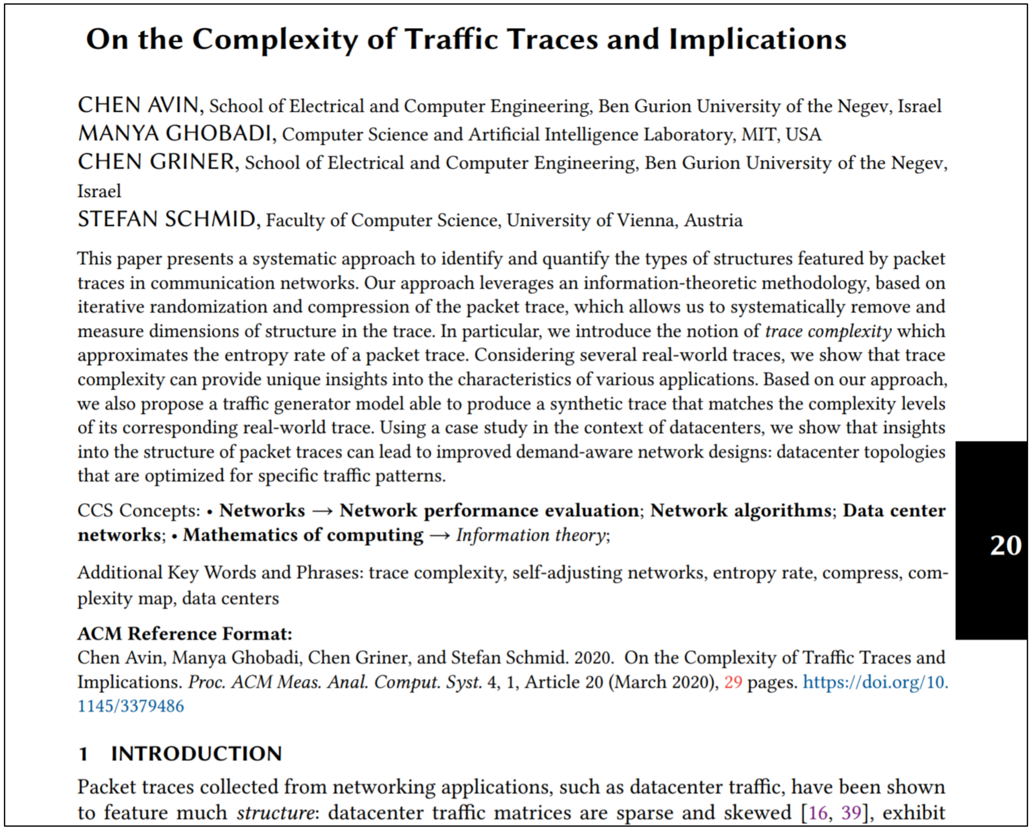


Potential gain!

Different structures!

Further Reading

# ACM SIGMETRICS 2020



## On the Complexity of Traffic Traces and Implications

CHEN AVIN, School of Electrical and Computer Engineering, Ben Gurion University of the Negev, Israel
MANYA GHOBADI, Computer Science and Artificial Intelligence Laboratory, MIT, USA
CHEN GRINER, School of Electrical and Computer Engineering, Ben Gurion University of the Negev, Israel
STEFAN SCHMID, Faculty of Computer Science, University of Vienna, Austria

This paper presents a systematic approach to identify and quantify the types of structures featured by packet traces in communication networks. Our approach leverages an information-theoretic methodology, based on iterative randomization and compression of the packet trace, which allows us to systematically remove and measure dimensions of structure in the trace. In particular, we introduce the notion of *trace complexity* which approximates the entropy rate of a packet trace. Considering several real-world traces, we show that trace complexity can provide unique insights into the characteristics of various applications. Based on our approach, we also propose a traffic generator model able to produce a synthetic trace that matches the complexity levels of its corresponding real-world trace. Using a case study in the context of datacenters, we show that insights into the structure of packet traces can lead to improved demand-aware network designs: datacenter topologies that are optimized for specific traffic patterns.

CCS Concepts: • **Networks** → **Network performance evaluation**; **Network algorithms**; **Data center networks**; • **Mathematics of computing** → *Information theory*;

Additional Key Words and Phrases: trace complexity, self-adjusting networks, entropy rate, compress, complexity map, data centers

**ACM Reference Format:**
Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid. 2020. On the Complexity of Traffic Traces and Implications. *Proc. ACM Meas. Anal. Comput. Syst.* 4, 1, Article 20 (March 2020), 29 pages. https://doi.org/10.1145/3379486

20

### 1 INTRODUCTION

Packet traces collected from networking applications, such as datacenter traffic, have been shown to feature much *structure*: datacenter traffic matrices are sparse and skewed [16, 39], exhibit

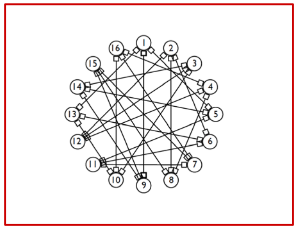
Question 2:

# How to Exploit Structure Algorithmically? Metrics for Achievable Efficiency?

Insight: Information-theoretic perspective useful here as well!

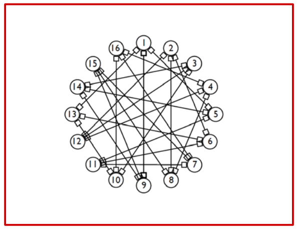
# Models and Connection to Datastructures & Coding

Traditional networks
(worst-case traffic)

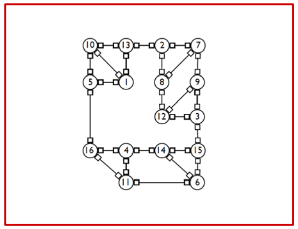


# Models and Connection to Datastructures & Coding

Traditional networks
(worst-case traffic)

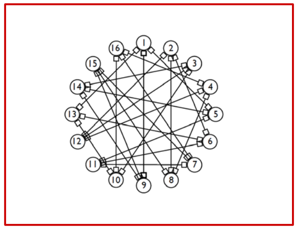


Demand-aware networks
(spatial structure)

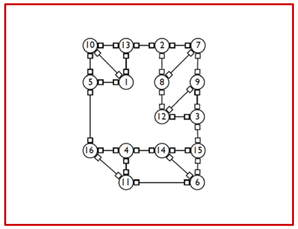


# Models and Connection to Datastructures & Coding

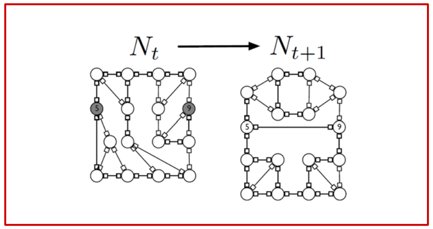
Traditional networks
(worst-case traffic)



Demand-aware networks
(spatial structure)

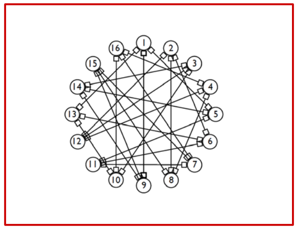


Self-adjusting networks
(temporal structure)

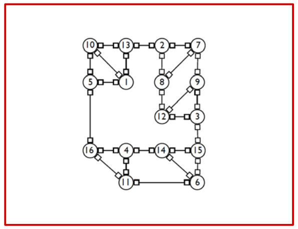


# Models and Connection to Datastructures & Coding

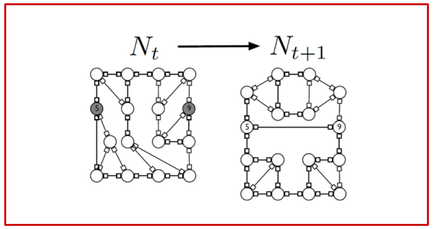
Traditional networks
(worst-case traffic)



Demand-aware networks
(spatial structure)



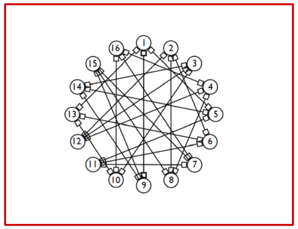
Self-adjusting networks
(temporal structure)



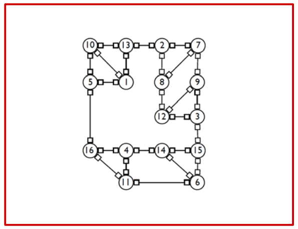
More structure: **lower routing cost**

# Models and Connection to Datastructures & Coding

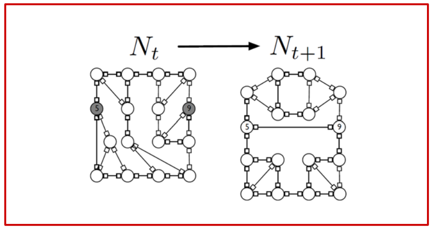
Traditional networks
(worst-case traffic)



Demand-aware networks
(spatial structure)

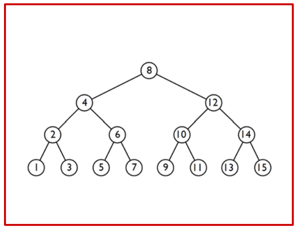


Self-adjusting networks
(temporal structure)

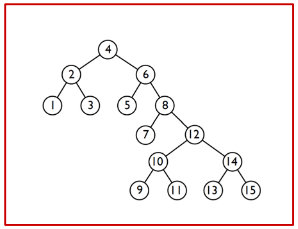


More structure: **lower routing cost**

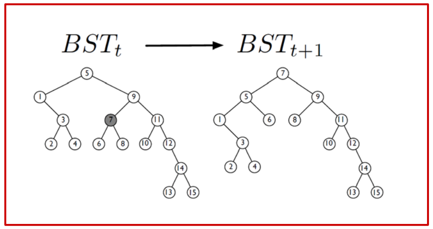
Traditional BST
(Worst-case coding)



Demand-aware BST
(Huffman coding)



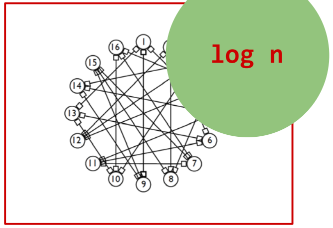
Self-adjusting BST
(Dynamic Huffman coding)



More structure: improved **access cost** / shorter **codes**

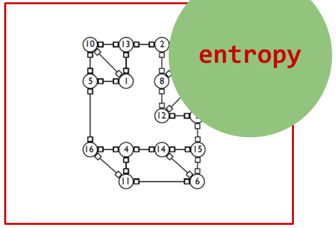
# Models and Connection to Datastructures & Coding

Traditional networks
(worst-case traffic)



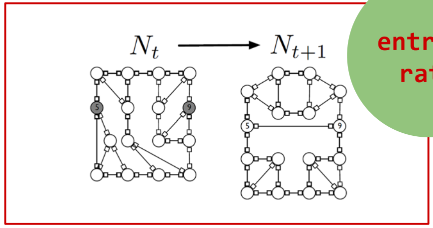
log n

Demand-aware networks
(spatial structure)



entropy

Self-adjusting networks
(temporal structure)

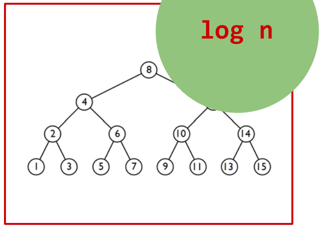


entropy rate

More than
an analogy!

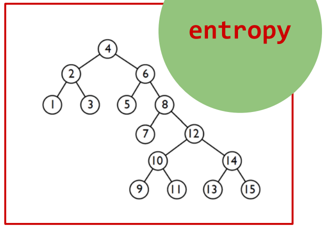
More structur lower routing cost

Traditional BST
(Worst-case



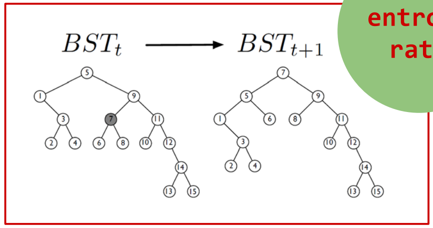
log n

Demand-aware BST
(Huffman c



entropy

Self-adjusting BST
(Dynamic Huffman codin



entropy rate

More structure: improved **access cost** / shorter **codes**

**Generalize methodology:**
**... and transfer entropy bounds and algorithms of data-structures to networks.**

**First result:**
**Demand-aware networks of asymptotically optimal route lengths.**

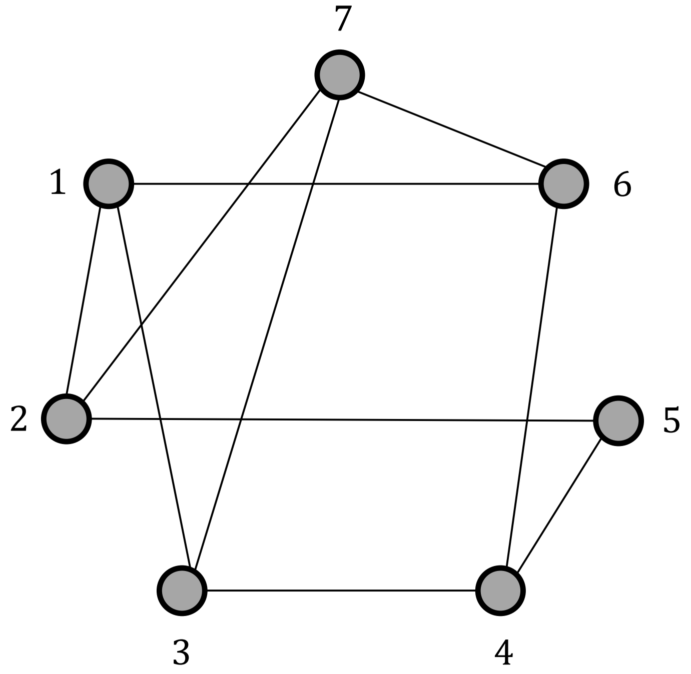
Case Study “Route Lengths”

# Constant-Degree Demand-Aware Network

Destinations

Sources

|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | $\frac{2}{65}$ | $\frac{1}{13}$ | $\frac{1}{65}$ | $\frac{1}{65}$ | $\frac{2}{65}$ | $\frac{3}{65}$ |
| 2 | $\frac{2}{65}$ | 0 | $\frac{1}{65}$ | 0 | 0 | 0 | $\frac{2}{65}$ |
| 3 | $\frac{1}{13}$ | $\frac{1}{65}$ | 0 | $\frac{2}{65}$ | 0 | 0 | $\frac{1}{13}$ |
| 4 | $\frac{1}{65}$ | 0 | $\frac{2}{65}$ | 0 | $\frac{4}{65}$ | 0 | 0 |
| 5 | $\frac{1}{65}$ | 0 | $\frac{3}{65}$ | $\frac{4}{65}$ | 0 | 0 | 0 |
| 6 | $\frac{2}{65}$ | 0 | 0 | 0 | 0 | 0 | $\frac{3}{65}$ |
| 7 | $\frac{3}{65}$ | $\frac{2}{65}$ | $\frac{1}{13}$ | 0 | 0 | $\frac{3}{65}$ | 0 |



$$\mathrm{ERL}(\mathcal{D},\mathrm{N}) = \sum_{(u,v)\in\mathcal{D}} p(u,v)\cdot d_N(u,v)$$

Case Study “Route Lengths”

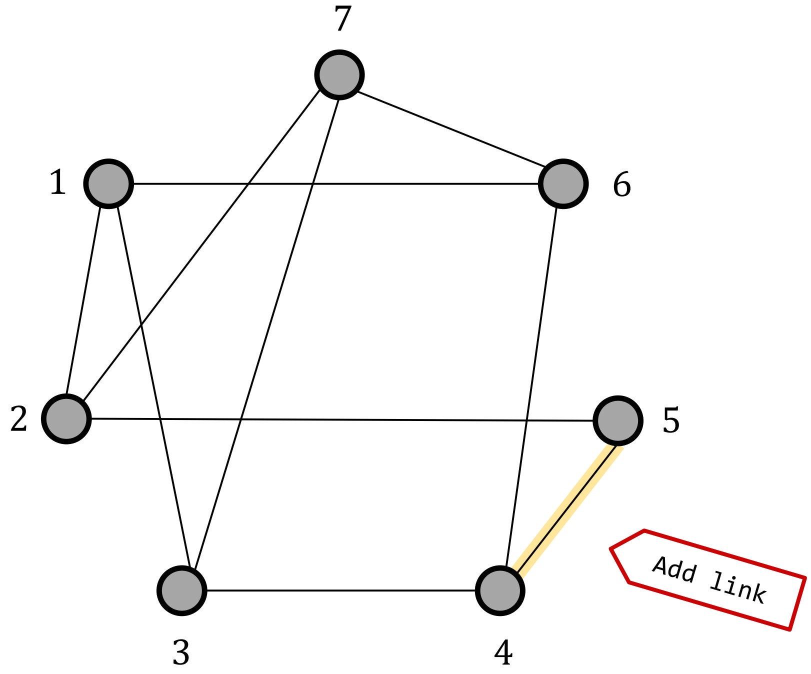
# Constant-Degree Demand-Aware Network

Destinations

Sources

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | $\frac{2}{65}$ | $\frac{1}{13}$ | $\frac{1}{65}$ | $\frac{1}{65}$ | $\frac{2}{65}$ | $\frac{3}{65}$ |
| 2 | $\frac{2}{65}$ | 0 | $\frac{1}{65}$ | 0 | 0 | 0 | $\frac{2}{65}$ |
| 3 | $\frac{1}{13}$ | $\frac{1}{65}$ | 0 | $\frac{2}{65}$ | 0 | 0 | $\frac{1}{13}$ |
| 4 | $\frac{1}{65}$ | 0 | $\frac{2}{65}$ | 0 | $\frac{4}{65}$ | 0 | 0 |
| 5 | $\frac{1}{65}$ | 0 | $\frac{3}{65}$ | [illegible] | 0 | 0 | 0 |
| 6 | $\frac{2}{65}$ | 0 | [illegible] | 0 | 0 | 0 | $\frac{3}{65}$ |
| 7 | $\frac{3}{65}$ | 2 [illegible] | $\frac{1}{13}$ | 0 | 0 | $\frac{3}{65}$ | 0 |

Much from 4 to 5



Add link

$$\mathrm{ERL}(\mathcal{D},\mathrm{N}) = \sum_{(u,v)\in\mathcal{D}} p(u,v)\cdot d_N(u,v)$$

Case Study “Route Lengths”

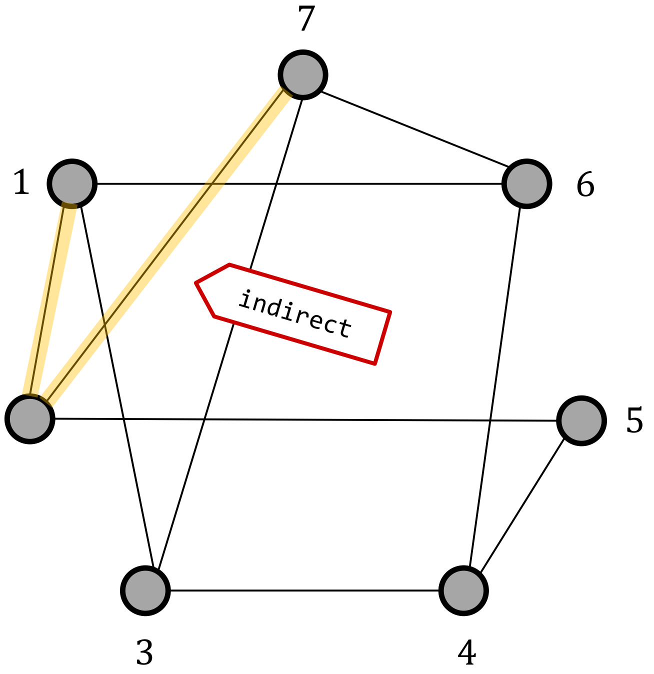
# Constant-Degree Demand-Aware Network

Communicated with many

Destinations

Sources

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | $\frac{2}{65}$ | $\frac{1}{13}$ | $\frac{1}{65}$ | $\frac{1}{65}$ | $\frac{2}{65}$ | $\frac{3}{65}$ |
| 2 | $\frac{2}{65}$ | 0 | $\frac{1}{65}$ | 0 | 0 | 0 | $\frac{2}{65}$ |
| 3 | $\frac{1}{13}$ | $\frac{1}{65}$ | 0 | $\frac{2}{65}$ | 0 | 0 | $\frac{1}{13}$ |
| 4 | $\frac{1}{65}$ | 0 | $\frac{2}{65}$ | 0 | $\frac{4}{65}$ | 0 | 0 |
| 5 | $\frac{1}{65}$ | 0 | $\frac{3}{65}$ | $\frac{4}{65}$ | 0 | 0 | 0 |
| 6 | $\frac{2}{65}$ | 0 | 0 | 0 | 0 | 0 | $\frac{3}{65}$ |
| 7 | $\frac{3}{65}$ | $\frac{2}{65}$ | $\frac{1}{13}$ | 0 | 0 | $\frac{3}{65}$ | 0 |

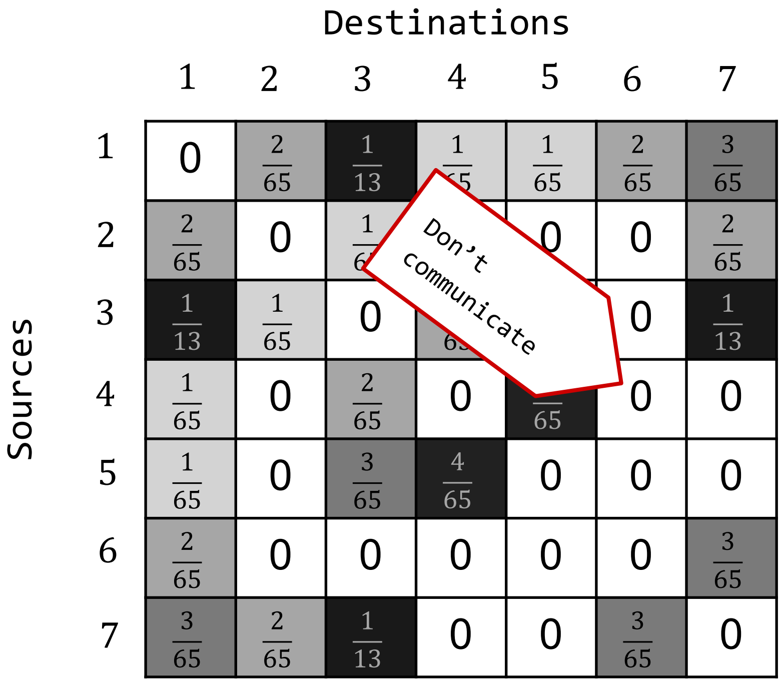


indirect

$$\mathrm{ERL}(\mathcal{D},\mathrm{N}) = \sum_{(u,v)\in\mathcal{D}} \mathrm{p}(u,v)\cdot \mathrm{d}_{\mathrm{N}}(u,v)$$

Case Study “Route Lengths”

# Constant-Degree Demand-Aware Network

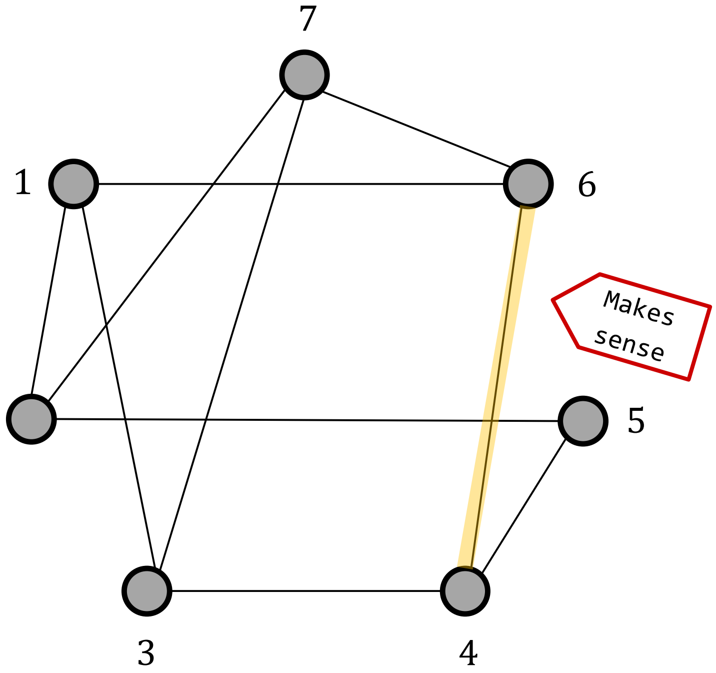


Destinations

Sources

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | $\frac{2}{65}$ | $\frac{1}{13}$ | $\frac{1}{65}$ | $\frac{1}{65}$ | $\frac{2}{65}$ | $\frac{3}{65}$ |
| 2 | $\frac{2}{65}$ | 0 | $\frac{1}{65}$ | [illegible] | 0 | 0 | $\frac{2}{65}$ |
| 3 | $\frac{1}{13}$ | $\frac{1}{65}$ | 0 | [illegible] | [illegible] | 0 | $\frac{1}{13}$ |
| 4 | $\frac{1}{65}$ | 0 | $\frac{2}{65}$ | 0 | [illegible] | 0 | 0 |
| 5 | $\frac{1}{65}$ | 0 | $\frac{3}{65}$ | $\frac{4}{65}$ | 0 | 0 | 0 |
| 6 | $\frac{2}{65}$ | 0 | 0 | 0 | 0 | 0 | $\frac{3}{65}$ |
| 7 | $\frac{3}{65}$ | $\frac{2}{65}$ | $\frac{1}{13}$ | 0 | 0 | $\frac{3}{65}$ | 0 |

Don’t communicate



Makes sense

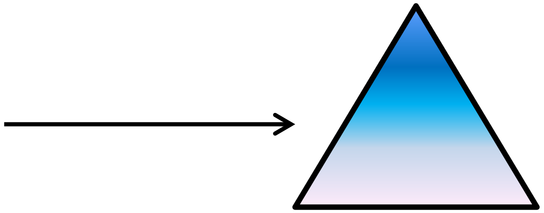
$$\mathrm{ERL}(\mathcal{D},N) = \sum_{(u,v)\in\mathcal{D}} p(u,v) \cdot d_N(u,v)$$

# Entropy Lower Bound

Destinations

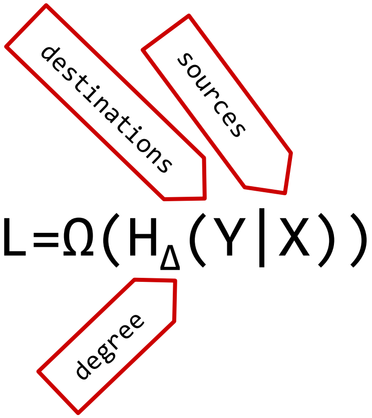
Sources

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | $\frac{2}{65}$ | $\frac{1}{13}$ | $\frac{1}{65}$ | $\frac{1}{65}$ | $\frac{2}{65}$ | $\frac{3}{65}$ |
| 2 | $\frac{2}{65}$ | 0 | $\frac{1}{65}$ | 0 | 0 | 0 | $\frac{2}{65}$ |
| 3 | $\frac{1}{13}$ | $\frac{1}{65}$ | 0 | $\frac{2}{65}$ | 0 | 0 | $\frac{1}{13}$ |
| 4 | $\frac{1}{65}$ | 0 | $\frac{2}{65}$ | 0 | $\frac{4}{65}$ | 0 | 0 |
| 5 | $\frac{1}{65}$ | 0 | $\frac{3}{65}$ | $\frac{4}{65}$ | 0 | 0 | 0 |
| 6 | $\frac{2}{65}$ | 0 | 0 | 0 | 0 | 0 | $\frac{3}{65}$ |
| | $\frac{3}{65}$ | $\frac{2}{65}$ | $\frac{1}{13}$ | 0 | 0 | $\frac{3}{65}$ | 0 |

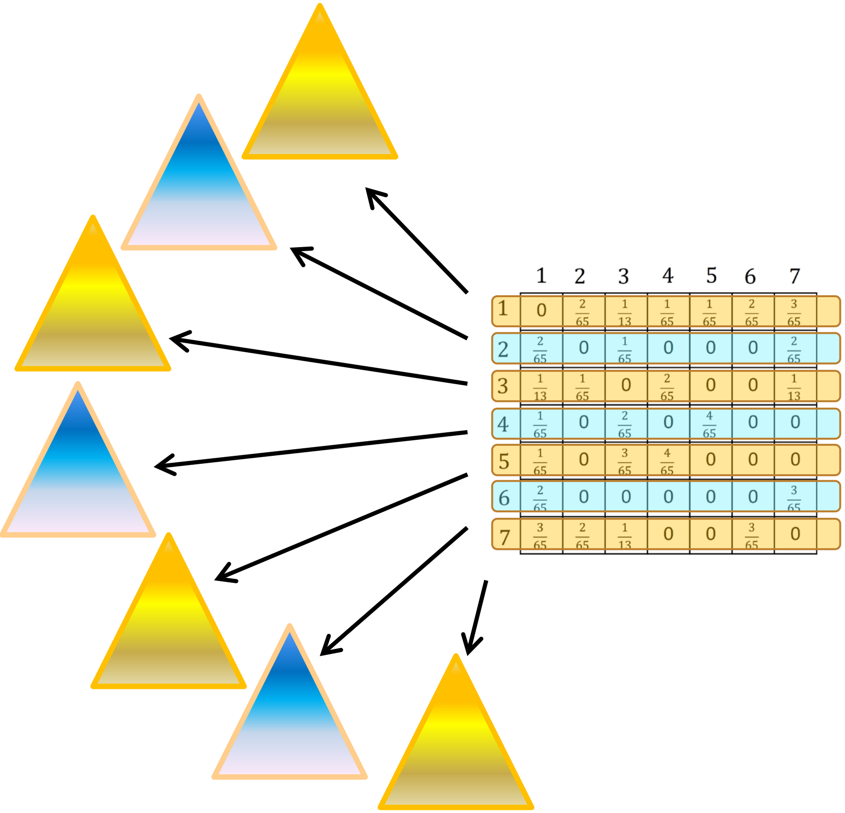


Huffman tree:
“**ego-tree**”

# Entropy Lower Bound



$$\mathrm{ERL}=\Omega(H_{\Delta}(Y|X))$$



|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | $\frac{2}{65}$ | $\frac{1}{13}$ | $\frac{1}{65}$ | $\frac{1}{65}$ | $\frac{2}{65}$ | $\frac{3}{65}$ |
| 2 | $\frac{2}{65}$ | 0 | $\frac{1}{65}$ | 0 | 0 | 0 | $\frac{2}{65}$ |
| 3 | $\frac{1}{13}$ | $\frac{1}{65}$ | 0 | $\frac{2}{65}$ | 0 | 0 | $\frac{1}{13}$ |
| 4 | $\frac{1}{65}$ | 0 | $\frac{2}{65}$ | 0 | $\frac{4}{65}$ | 0 | 0 |
| 5 | $\frac{1}{65}$ | 0 | $\frac{3}{65}$ | $\frac{4}{65}$ | 0 | 0 | 0 |
| 6 | $\frac{2}{65}$ | 0 | 0 | 0 | 0 | 0 | $\frac{3}{65}$ |
| 7 | $\frac{3}{65}$ | $\frac{2}{65}$ | $\frac{1}{13}$ | 0 | 0 | $\frac{3}{65}$ | 0 |

# Entropy Upper Bound

⇢ Idea for algorithm:

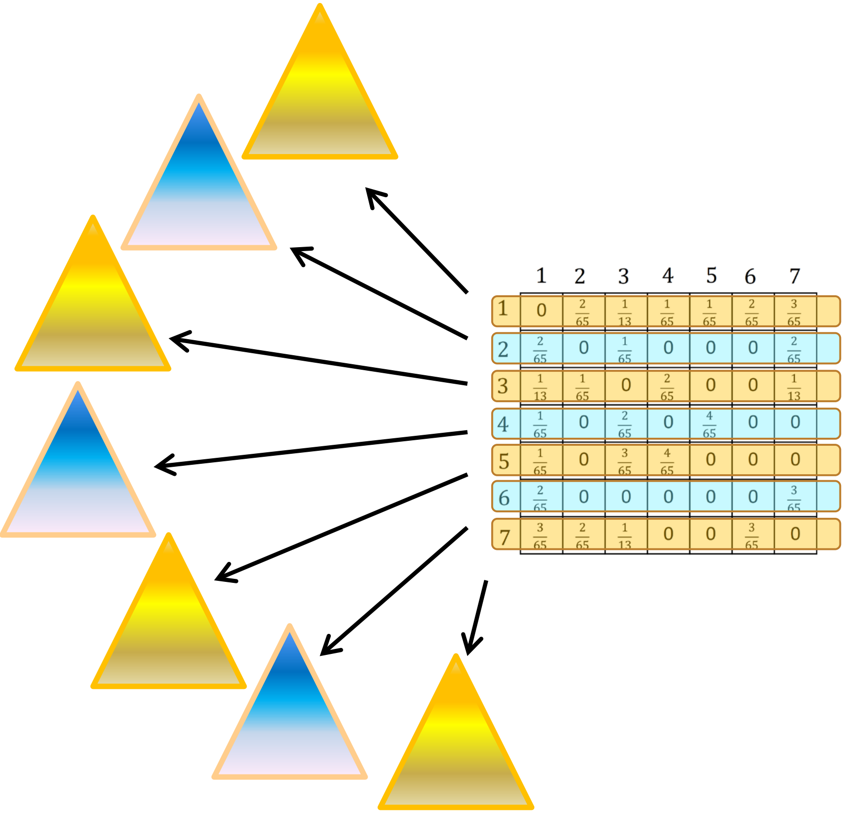
- ⇀ union of trees
- ⇀ reduce degree
- ⇀ but keep distances

⇢ Ok for sparse demands

- ⇀ not everyone gets tree
- ⇀ helper nodes

Static

What about dynamic case?

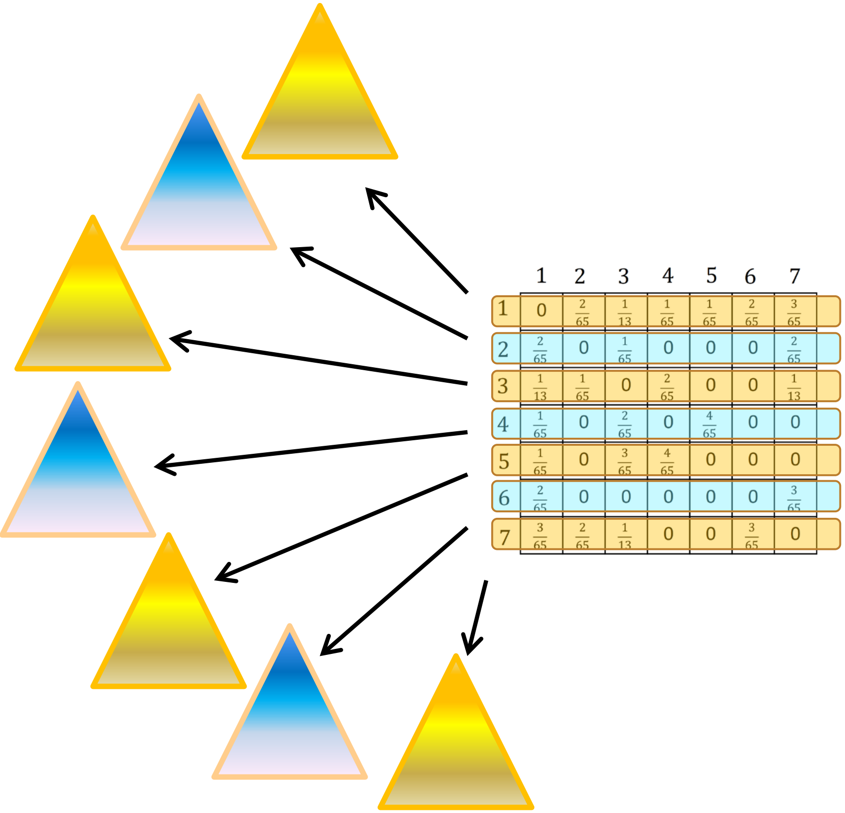


|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | $\frac{2}{65}$ | $\frac{1}{13}$ | $\frac{1}{65}$ | $\frac{1}{65}$ | $\frac{2}{65}$ | $\frac{3}{65}$ |
| 2 | $\frac{2}{65}$ | 0 | $\frac{1}{65}$ | 0 | 0 | 0 | $\frac{2}{65}$ |
| 3 | $\frac{1}{13}$ | $\frac{1}{65}$ | 0 | $\frac{2}{65}$ | 0 | 0 | $\frac{1}{13}$ |
| 4 | $\frac{1}{65}$ | 0 | $\frac{2}{65}$ | 0 | $\frac{4}{65}$ | 0 | 0 |
| 5 | $\frac{1}{65}$ | 0 | $\frac{3}{65}$ | $\frac{4}{65}$ | 0 | 0 | 0 |
| 6 | $\frac{2}{65}$ | 0 | 0 | 0 | 0 | 0 | $\frac{3}{65}$ |
| 7 | $\frac{3}{65}$ | $\frac{2}{65}$ | $\frac{1}{13}$ | 0 | 0 | $\frac{3}{65}$ | 0 |

# Dynamic Setting

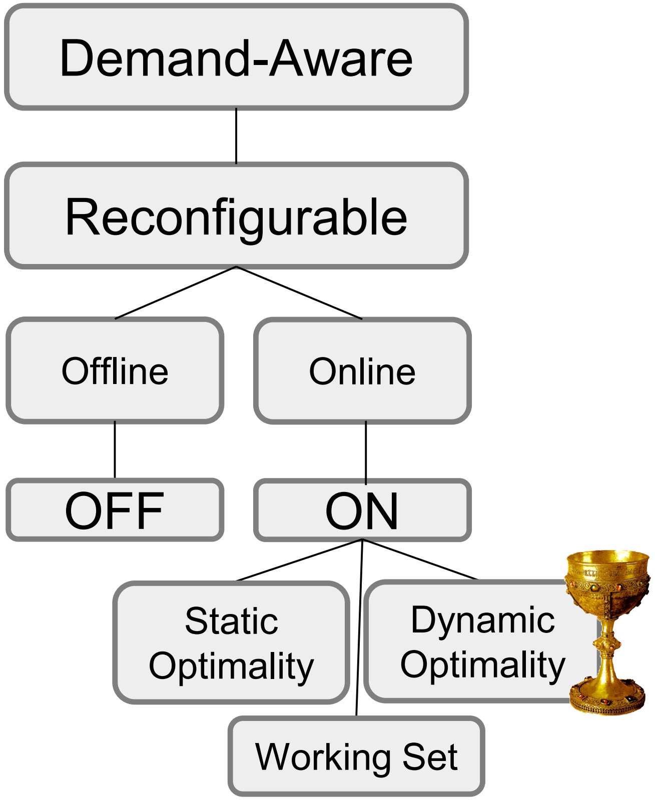
- Dynamic the same:
  - union of **dynamic ego-trees**
- E.g., SplayNets
- Online algorithms

**Dynamic**



|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | $\frac{2}{65}$ | $\frac{1}{13}$ | $\frac{1}{65}$ | $\frac{1}{65}$ | $\frac{2}{65}$ | $\frac{3}{65}$ |
| 2 | $\frac{2}{65}$ | 0 | $\frac{1}{65}$ | 0 | 0 | 0 | $\frac{2}{65}$ |
| 3 | $\frac{1}{13}$ | $\frac{1}{65}$ | 0 | $\frac{2}{65}$ | 0 | 0 | $\frac{1}{13}$ |
| 4 | $\frac{1}{65}$ | 0 | $\frac{2}{65}$ | 0 | $\frac{4}{65}$ | 0 | 0 |
| 5 | $\frac{1}{65}$ | 0 | $\frac{3}{65}$ | $\frac{4}{65}$ | 0 | 0 | 0 |
| 6 | $\frac{2}{65}$ | 0 | 0 | 0 | 0 | 0 | $\frac{3}{65}$ |
| 7 | $\frac{3}{65}$ | $\frac{2}{65}$ | $\frac{1}{13}$ | 0 | 0 | $\frac{3}{65}$ | 0 |

# Dynamic Objectives



Demand-Aware

Reconfigurable

Offline

Online

OFF

ON

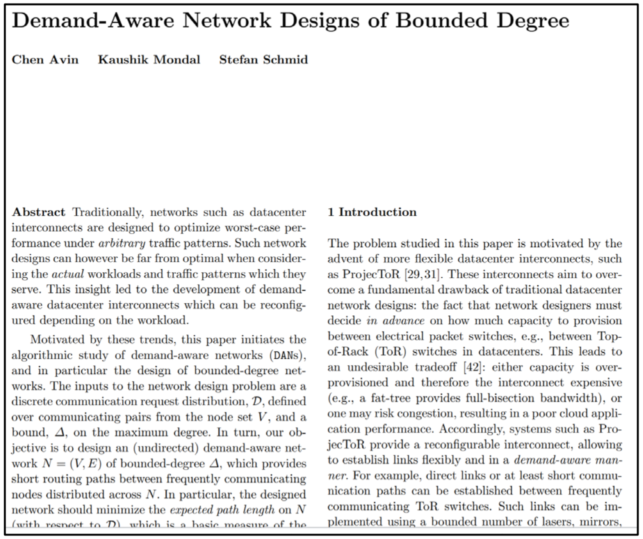
Static Optimality

Dynamic Optimality

Working Set

# Further Reading

Static DAN



## Demand-Aware Network Designs of Bounded Degree

**Chen Avin  Kaushik Mondal  Stefan Schmid**

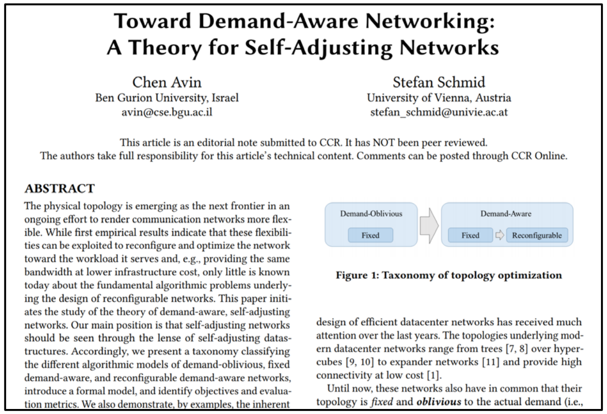
**Abstract** Traditionally, networks such as datacenter interconnects are designed to optimize worst-case performance under *arbitrary* traffic patterns. Such network designs can however be far from optimal when considering the *actual* workloads and traffic patterns which they serve. This insight led to the development of demand-aware datacenter interconnects which can be reconfigured depending on the workload.

Motivated by these trends, this paper initiates the algorithmic study of demand-aware networks (DANs), and in particular the design of bounded-degree networks. The inputs to the network design problem are a discrete communication request distribution, $\mathcal{D}$, defined over communicating pairs from the node set $V$, and a bound, $\Delta$, on the maximum degree. In turn, our objective is to design an (undirected) demand-aware network $N = (V, E)$ of bounded-degree $\Delta$, which provides short routing paths between frequently communicating nodes distributed across $N$. In particular, the designed network should minimize the *expected path length* on $N$ (with respect to $\mathcal{D}$) which is a basic measure of the

**1 Introduction**

The problem studied in this paper is motivated by the advent of more flexible datacenter interconnects, such as ProjecToR [29,31]. These interconnects aim to overcome a fundamental drawback of traditional datacenter network designs: the fact that network designers must decide *in advance* on how much capacity to provision between electrical packet switches, e.g., between Top-of-Rack (ToR) switches in datacenters. This leads to an undesirable tradeoff [42]: either capacity is over-provisioned and therefore the interconnect expensive (e.g., a fat-tree provides full-bisection bandwidth), or one may risk congestion, resulting in a poor cloud application performance. Accordingly, systems such as ProjecToR provide a reconfigurable interconnect, allowing to establish links flexibly and in a *demand-aware manner*. For example, direct links or at least short communication paths can be established between frequently communicating ToR switches. Such links can be implemented using a bounded number of lasers, mirrors,

Overview: Models



## Toward Demand-Aware Networking: A Theory for Self-Adjusting Networks

Chen Avin
Ben Gurion University, Israel
avin@cse.bgu.ac.il

Stefan Schmid
University of Vienna, Austria
stefan_schmid@univie.ac.at

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

**ABSTRACT**

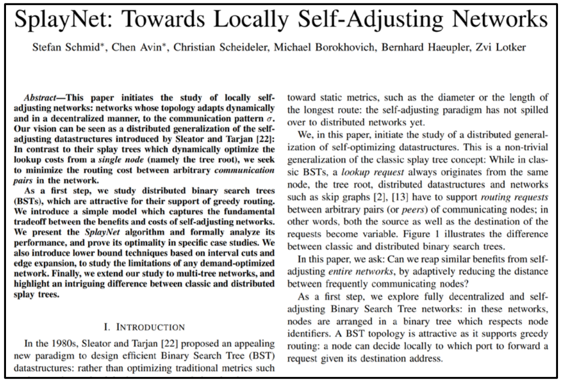
The physical topology is emerging as the next frontier in an ongoing effort to render communication networks more flexible. While first empirical results indicate that these flexibilities can be exploited to reconfigure and optimize the network toward the workload it serves and, e.g., providing the same bandwidth at lower infrastructure cost, only little is known today about the fundamental algorithmic problems underlying the design of reconfigurable networks. This paper initiates the study of the theory of demand-aware, self-adjusting networks. Our main position is that self-adjusting networks should be seen through the lense of self-adjusting datastructures. Accordingly, we present a taxonomy classifying the different algorithmic models of demand-oblivious, fixed demand-aware, and reconfigurable demand-aware networks, introduce a formal model, and identify objectives and evaluation metrics. We also demonstrate, by examples, the inherent

**Figure 1: Taxonomy of topology optimization**

design of efficient datacenter networks has received much attention over the last years. The topologies underlying modern datacenter networks range from trees [7, 8] over hypercubes [9, 10] to expander networks [11] and provide high connectivity at low cost [1].

Until now, these networks also have in common that their topology is *fixed* and ***oblivious*** to the actual demand (i.e.,

Dynamic DAN



## SplayNet: Towards Locally Self-Adjusting Networks

Stefan Schmid*, Chen Avin*, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, Zvi Lotker

***Abstract*—This paper initiates the study of locally self-adjusting networks: networks whose topology adapts dynamically and in a decentralized manner, to the communication pattern $\sigma$. Our vision can be seen as a distributed generalization of the self-adjusting datastructures introduced by Sleator and Tarjan [22]: In contrast to their splay trees which dynamically optimize the lookup costs from a *single node* (namely the tree root), we seek to minimize the routing cost between arbitrary *communication pairs* in the network.**

**As a first step, we study distributed binary search trees (BSTs), which are attractive for their support of greedy routing. We introduce a simple model which captures the fundamental tradeoff between the benefits and costs of self-adjusting networks. We present the *SplayNet* algorithm and formally analyze its performance, and prove its optimality in specific case studies. We also introduce lower bound techniques based on interval cuts and edge expansion, to study the limitations of any demand-optimized network. Finally, we extend our study to multi-tree networks, and highlight an intriguing difference between classic and distributed splay trees.**

### I. Introduction

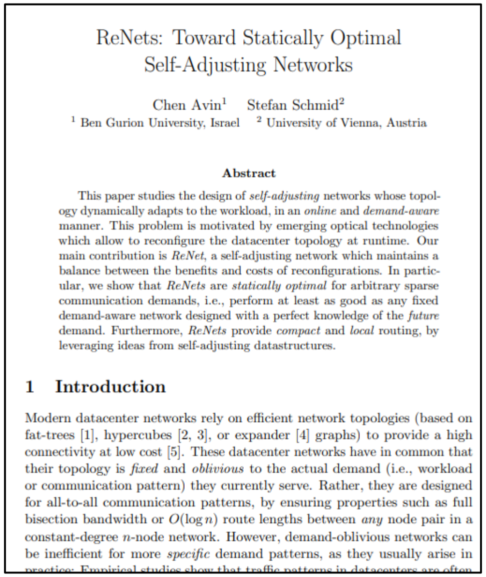
In the 1980s, Sleator and Tarjan [22] proposed an appealing new paradigm to design efficient Binary Search Tree (BST) datastructures: rather than optimizing traditional metrics such toward static metrics, such as the diameter or the length of the longest route: the self-adjusting paradigm has not spilled over to distributed networks yet.

We, in this paper, initiate the study of a distributed generalization of self-optimizing datastructures. This is a non-trivial generalization of the classic splay tree concept: While in classic BSTs, a *lookup request* always originates from the same node, the tree root, distributed datastructures and networks such as skip graphs [2], [13] have to support *routing requests* between arbitrary pairs (or *peers*) of communicating nodes; in other words, both the source as well as the destination of the requests become variable. Figure 1 illustrates the difference between classic and distributed binary search trees.

In this paper, we ask: Can we reap similar benefits from self-adjusting *entire networks*, by adaptively reducing the distance between frequently communicating nodes?

As a first step, we explore fully decentralized and self-adjusting Binary Search Tree networks: in these networks, nodes are arranged in a binary tree which respects node identifiers. A BST topology is attractive as it supports greedy routing: a node can decide locally to which port to forward a request given its destination address.

Static Optimality



## ReNets: Toward Statically Optimal Self-Adjusting Networks

Chen Avin[1]  Stefan Schmid[2]
[1] Ben Gurion University, Israel  [2] University of Vienna, Austria

**Abstract**

This paper studies the design of *self-adjusting* networks whose topology dynamically adapts to the workload, in an *online* and *demand-aware* manner. This problem is motivated by emerging optical technologies which allow to reconfigure the datacenter topology at runtime. Our main contribution is *ReNet*, a self-adjusting network which maintains a balance between the benefits and costs of reconfigurations. In particular, we show that *ReNets* are *statically optimal* for arbitrary sparse communication demands, i.e., perform at least as good as any fixed demand-aware network designed with a perfect knowledge of the *future* demand. Furthermore, *ReNets* provide *compact* and *local* routing, by leveraging ideas from self-adjusting datastructures.

### 1 Introduction

Modern datacenter networks rely on efficient network topologies (based on fat-trees [1], hypercubes [2, 3], or expander [4] graphs) to provide a high connectivity at low cost [5]. These datacenter networks have in common that their topology is *fixed* and *oblivious* to the actual demand (i.e., workload or communication pattern) they currently serve. Rather, they are designed for all-to-all communication patterns, by ensuring properties such as full bisection bandwidth or $O(\log n)$ route lengths between *any* node pair in a constant-degree $n$-node network. However, demand-oblivious networks can be inefficient for more *specific* demand patterns, as they usually arise in practice: Empirical studies show that traffic patterns in datacenters are often

# Future Work:
# Models, Metrics, Algos

so far

scratched surface

to do ☺

Notion of self-adjusting networks opens a **large uncharted field** with many questions:

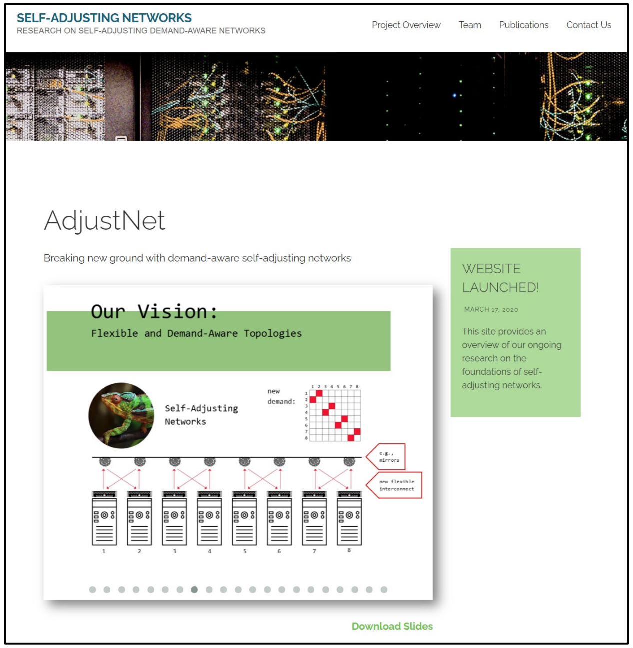
⇀ Metrics and algorithms: by how much can load be lowered, **energy** reduced, quality-of-service improved, etc. in demand-aware networks? Even for **route length** not clear!

⇀ How to **model** reconfiguration costs?

⇀ Impact on **other layers**?

**Requires knowledge in networking, distributed systems, algorithms, performance evaluation.**

# Websites



**SELF-ADJUSTING NETWORKS**
RESEARCH ON SELF-ADJUSTING DEMAND-AWARE NETWORKS

## AdjustNet

Breaking new ground with demand-aware self-adjusting networks

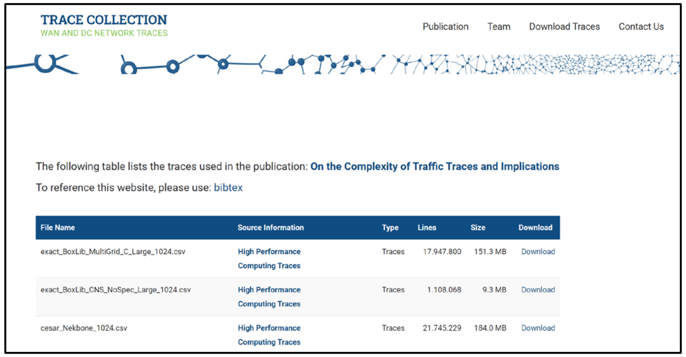
**Our Vision:**
Flexible and Demand-Aware Topologies

WEBSITE LAUNCHED!

MARCH 17, 2020

This site provides an overview of our ongoing research on the foundations of self-adjusting networks.

**http://self-adjusting.net/**
**Project website**



**TRACE COLLECTION**
WAN AND DC NETWORK TRACES

The following table lists the traces used in the publication: **On the Complexity of Traffic Traces and Implications**

To reference this website, please use: bibtex

| File Name | Source Information | Type | Lines | Size | Download |
|---|---|---|---|---|---|
| exact_BoxLib_MultiGrid_C_Large_1024.csv | High Performance Computing Traces | Traces | 17.947.800 | 151.3 MB | Download |
| exact_BoxLib_CNS_NoSpec_Large_1024.csv | High Performance Computing Traces | Traces | 1.108.068 | 9.3 MB | Download |
| cesar_Nekbone_1024.csv | High Performance Computing Traces | Traces | 21.745.229 | 184.0 MB | Download |

**https://trace-collection.net/**
**Trace collection website**

# Selected References

**On the Complexity of Traffic Traces and Implications**
Chen Avin, Manya Ghobadi, Chen Griner, and Stefan Schmid.
ACM SIGMETRICS, Boston, Massachusetts, USA, June 2020.

**Survey of Reconfigurable Data Center Networks: Enablers, Algorithms, Complexity**
Klaus-Tycho Foerster and Stefan Schmid.
**SIGACT News**, June 2019.

**Toward Demand-Aware Networking: A Theory for Self-Adjusting Networks (Editorial)**
Chen Avin and Stefan Schmid.
ACM SIGCOMM Computer Communication Review (**CCR**), October 2018.

**Measuring the Complexity of Network Traffic Traces**
Chen Griner, Chen Avin, Manya Ghobadi, and Stefan Schmid.
arXiv, 2019.

**Demand-Aware Network Design with Minimal Congestion and Route Lengths**
Chen Avin, Kaushik Mondal, and Stefan Schmid.
38th IEEE Conference on Computer Communications (**INFOCOM**), Paris, France, April 2019.

**Distributed Self-Adjusting Tree Networks**
Bruna Peres, Otavio Augusto de Oliveira Souza, Olga Goussevskaia, Chen Avin, and Stefan Schmid.
38th IEEE Conference on Computer Communications (**INFOCOM**), Paris, France, April 2019.

**Efficient Non-Segregated Routing for Reconfigurable Demand-Aware Networks**
Thomas Fenz, Klaus-Tycho Foerster, Stefan Schmid, and Anaïs Villedieu.
**IFIP Networking**, Warsaw, Poland, May 2019.

**DaRTree: Deadline-Aware Multicast Transfers in Reconfigurable Wide-Area Networks**
Long Luo, Klaus-Tycho Foerster, Stefan Schmid, and Hongfang Yu.
IEEE/ACM International Symposium on Quality of Service (**IWQoS**), Phoenix, Arizona, USA, June 2019.

**Demand-Aware Network Designs of Bounded Degree**
Chen Avin, Kaushik Mondal, and Stefan Schmid.
31st International Symposium on Distributed Computing (**DISC**), Vienna, Austria, October 2017.

**SplayNet: Towards Locally Self-Adjusting Networks**
Stefan Schmid, Chen Avin, Christian Scheideler, Michael Borokhovich, Bernhard Haeupler, and Zvi Lotker.
IEEE/ACM Transactions on Networking (**TON**), Volume 24, Issue 3, 2016. Early version: IEEE **IPDPS** 2013.

**Characterizing the Algorithmic Complexity of Reconfigurable Data Center Architectures**
Klaus-Tycho Foerster, Monia Ghobadi, and Stefan Schmid.
ACM/IEEE Symposium on Architectures for Networking and Communications Systems (**ANCS**), Ithaca, New York, USA, July 2018.